BRASS: Building Resource Adaptive Software Systems

Suresh Jagannathan, I2O

April 8, 2015

Agenda

1230 – 1300	Check-In/Registration	
1300 – 1315	Welcome	Suresh Jagannathan, DARPA
1315 – 1345	Contracts	Michael Blackstone, DARPA
1345 – 1430	BRASS Overview	Suresh Jagannathan, DARPA
1430 – 1500	Attendee Presentations	
1500 – 1600	Break	
1600 – 1700	Government Response to Questions	Suresh Jagannathan, DARPA

Logistics

- DARPA-BAA 15-36
  - FedBizOpps website (http://www.fedbizopps.gov)
  - Grants.gov website (http://www.grants.gov)
  - Posting Date: April 7, 2015
  - Proposal Due Date: May 22, 2015 at Noon ET

- Attendee Presentations
  - Two minutes in length and a single content slide
  - Slides are pre-loaded on laptop

- Government Response to Questions Session
  - Questions can be submitted until 3:30pm to BRASS@darpa.mil
  - Questions will be answered during Q&A session

- Online Material
  - Program Page: http://www.darpa.mil/Our_Work/I2O/Programs/Building_Resource_Adaptive_Software_Systems_(BRASS).aspx
    - Copy of Government Presentations
    - Video of BRASS Overview Presentation
    - Frequently Asked Questions (FAQs)
  - Teaming site: https://www.schafertmd.com/darpa/i2o/brass/teaming/

Understanding an application's environment

Abstractions

Relax: expose timing faults to software to improve error tolerance with lower power

Green: Online monitoring and QoS for approximate computing of loops and functions

EnerJ: Type system that isolates parts of the program that must be precise from those that can be approximate

PetaBricks: Autotuning abstractions that extends notions of algorithmic choice with support for selection of variable accuracy algorithms

Coccinelle: A DSL for describing semantic patches in collateral evolution of Linux device drivers

Self-adjusting computation: Identify notion of changeable computations that respond to changes in input

Bi-directional programming and lenses: Maintain semantic consistency between related sources of information

Program

Pyxsis: Offloading computation from Java applications to SQL database server for improved performance

RAML: A first-order dialect of ML that supports a multivariate amortized analysis that computes polynomial resource bounds

Terminator and T2: Automatic verification tool for proving termination and liveness properties in CTL

Resynth: Automated refactoring tool using program synthesis

Cache and I/O Efficient Functional Algorithms: A cost model for analyzing the memory efficiency of algorithms expressed in a functional language

CodePhage: Automatic transfer and patching of correct code from donor applications intro recipient applications to eliminate errors

Monitoring, Instrumentation, Dynamic Translation, VMs

HalVM and MirageOS: Haskell and Ocaml-based unikernels running on Xen

Digital Vellum: Cloud storage in which every file is bundled with information

Drawbridge: Virtualization support for application sandboxing using a library OS

Lancet: Surgical precision JIT compiler with metaprogramming support

Kitsune: A dynamic software update system for C capable of whole-program updates

REFRACT: Failure avoidance through adaptive reconfiguration

Naid: Timely dataflow as a computational model for scalable Big Data analytics of changing data at interactive timescales

Symbolic Execution

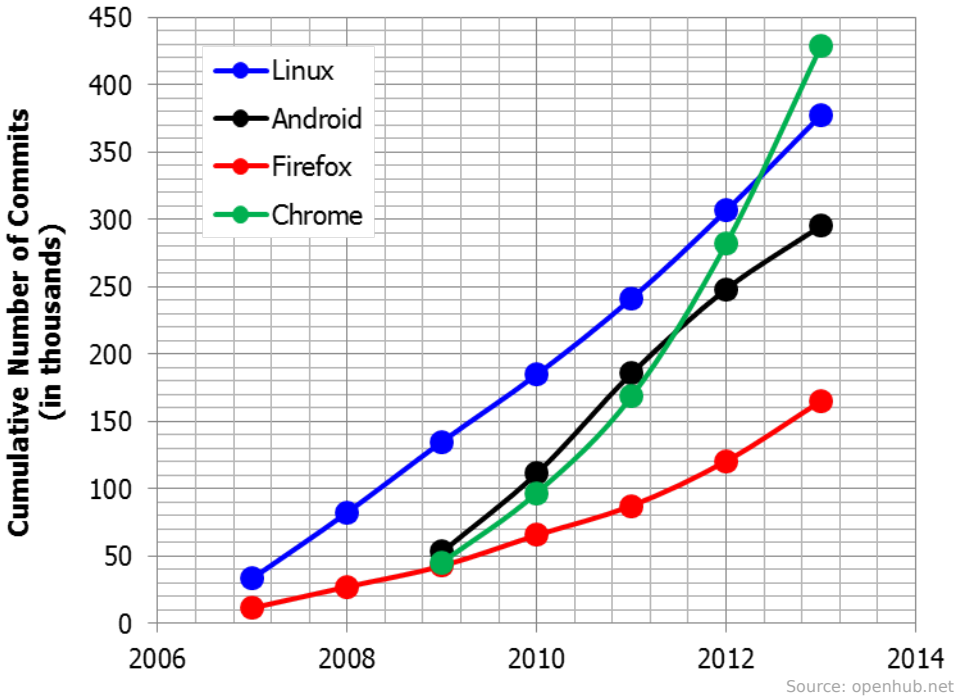
KLEE: Symbolic execution engine to generate high-coverage test cases

S2E: Scalable path-sensitive platform

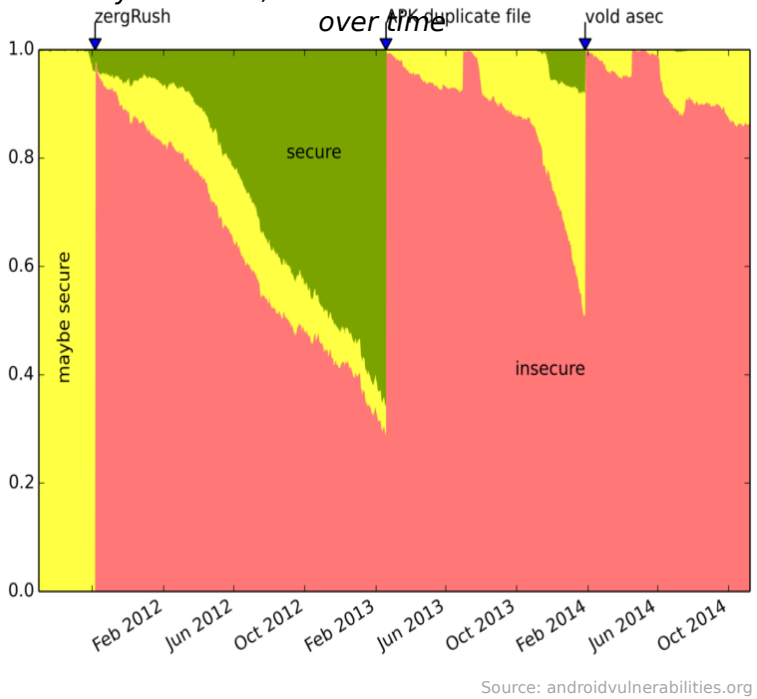
Types, Semantics, Verification

Battery Transition Systems: Formal model of energy systems with recovery effect behavior	*CompcertX*: Language-based account of abstraction layers and implementation independence properties
Session Types: Type systems for expressing communication protocols	*Data description languages*: DSLs and type systems for expressing the structure and representation of *ad hoc* data formats
CALM: Consistency as logical monotonicity for reasoning about eventual consistency	*Separation Logic*: Program logic for verifying properties of dynamic data structures and the heap
Vault: Linear type system to track...	*Assume Guarantee Reasoning*: Automated compositional

Software Change and Vulnerability



Proportion of Android devices running insecure, maybe secure, and secure versions of Android over time



Source: androidvulnerabilities.org

Past 12 Months		
	Files Modified (in thousands)	**Lines Modified (in millions)**
Linux	34	82
Android	80	26
Firefox	83	50
Chrome	120	23

Source: openhub.net

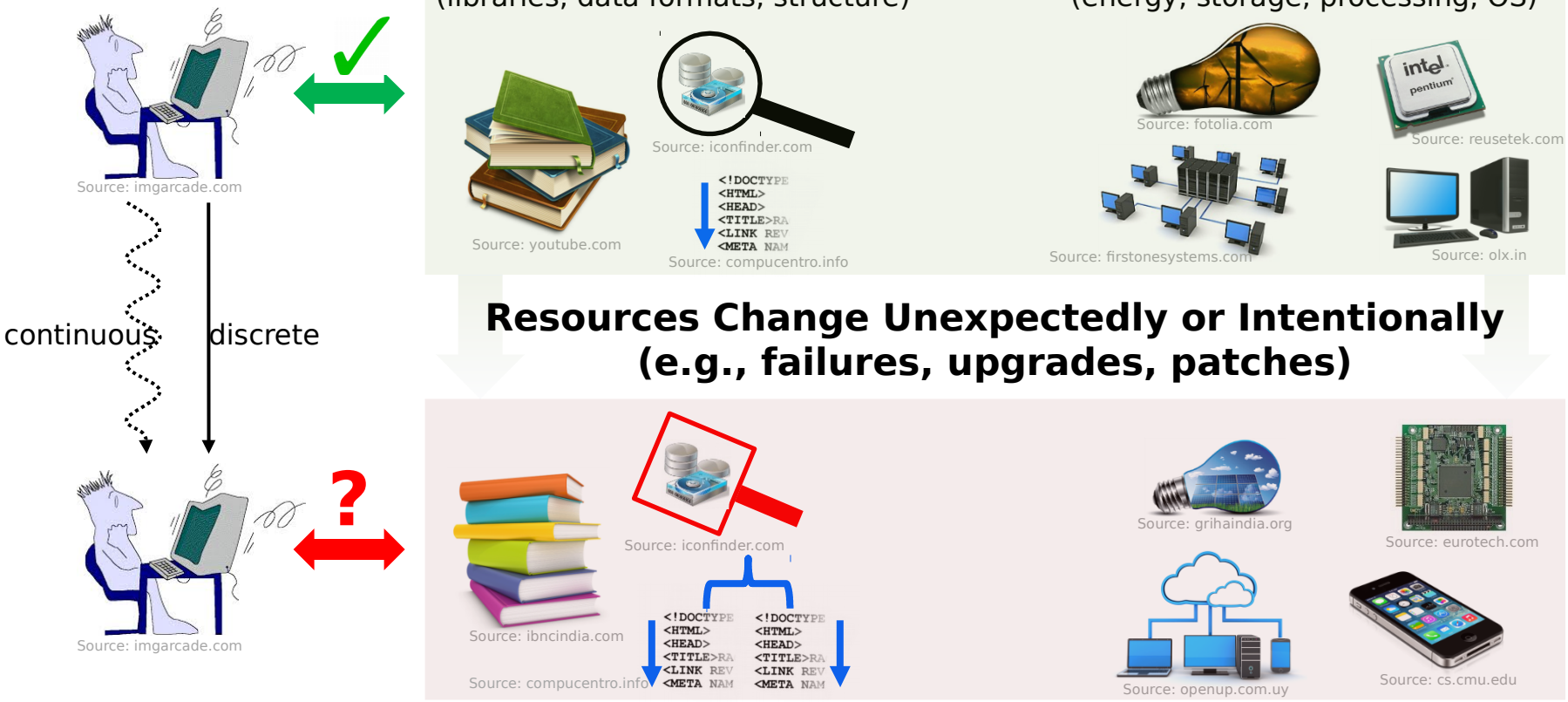
Nearly every U.S. weapons program tested in fiscal 2014 showed “significant vulnerabilities” to cyber attacks, including **misconfigured, unpatched and outdated software**, the Pentagon’s chief weapons tester said in his annual report.

- Reuters.com, Jan. 2015

“The [in]ability to preserve and run software over long periods of time may make the 21st century an information black hole.”

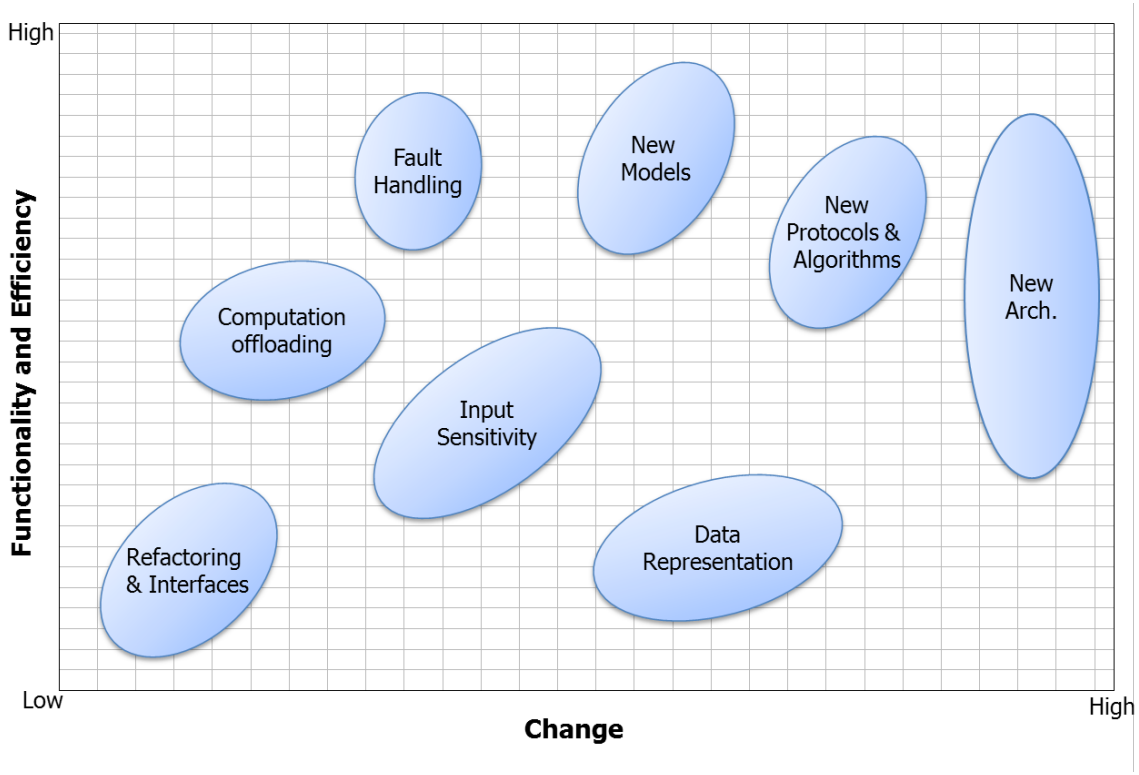
- Google VP Vint Cerf, Feb. 2015

Problem



Functionally correct code may operate incorrectly or inefficiently, as the operational environment in which it executes changes

Functionality and Change



Restricted Functionality

- Physical resources (energy, memory, network, communication)
- Restricted policies (security)

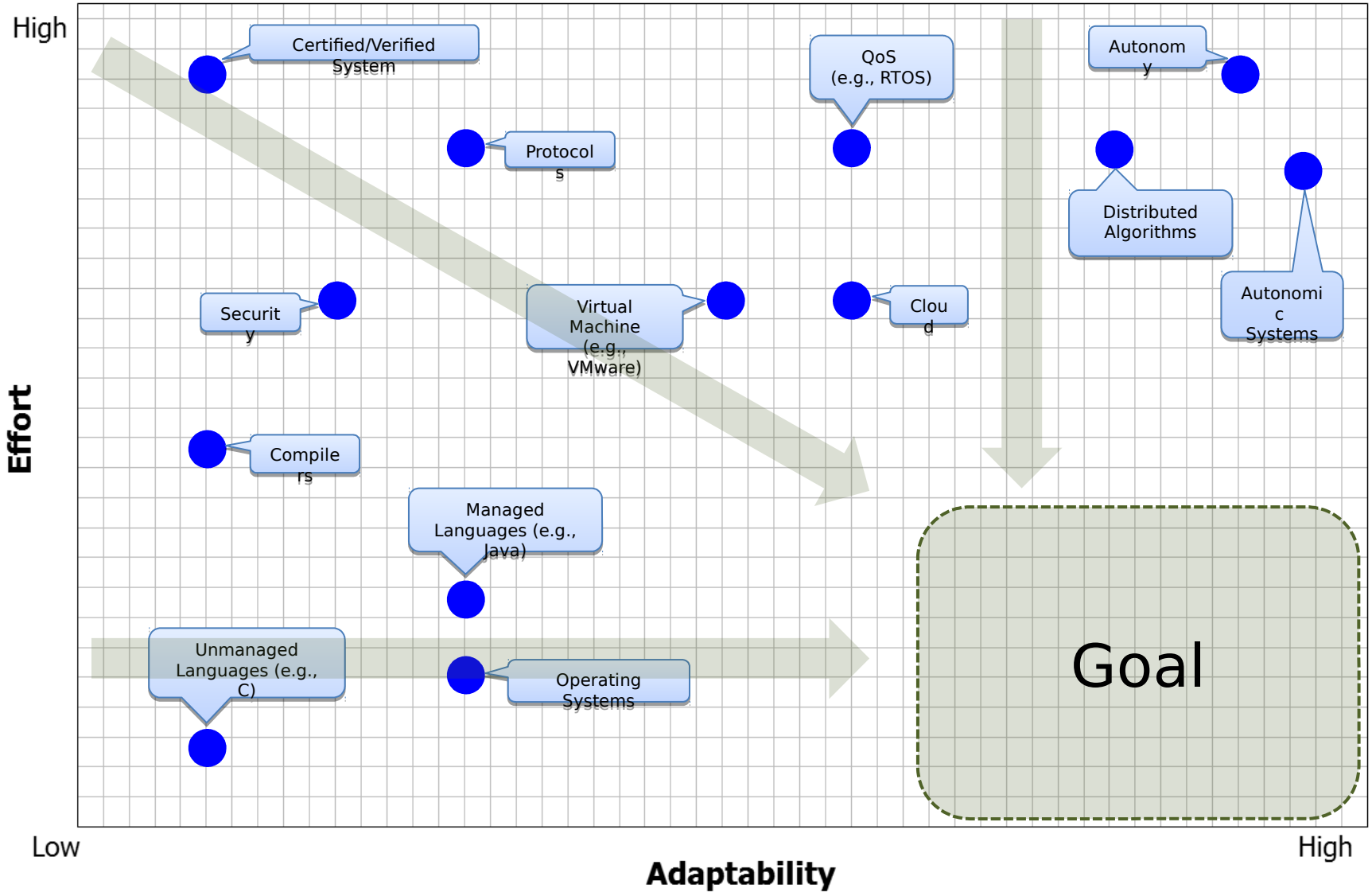
Enhanced Functionality

- Protocols and runtime services
- Cross-language interoperability

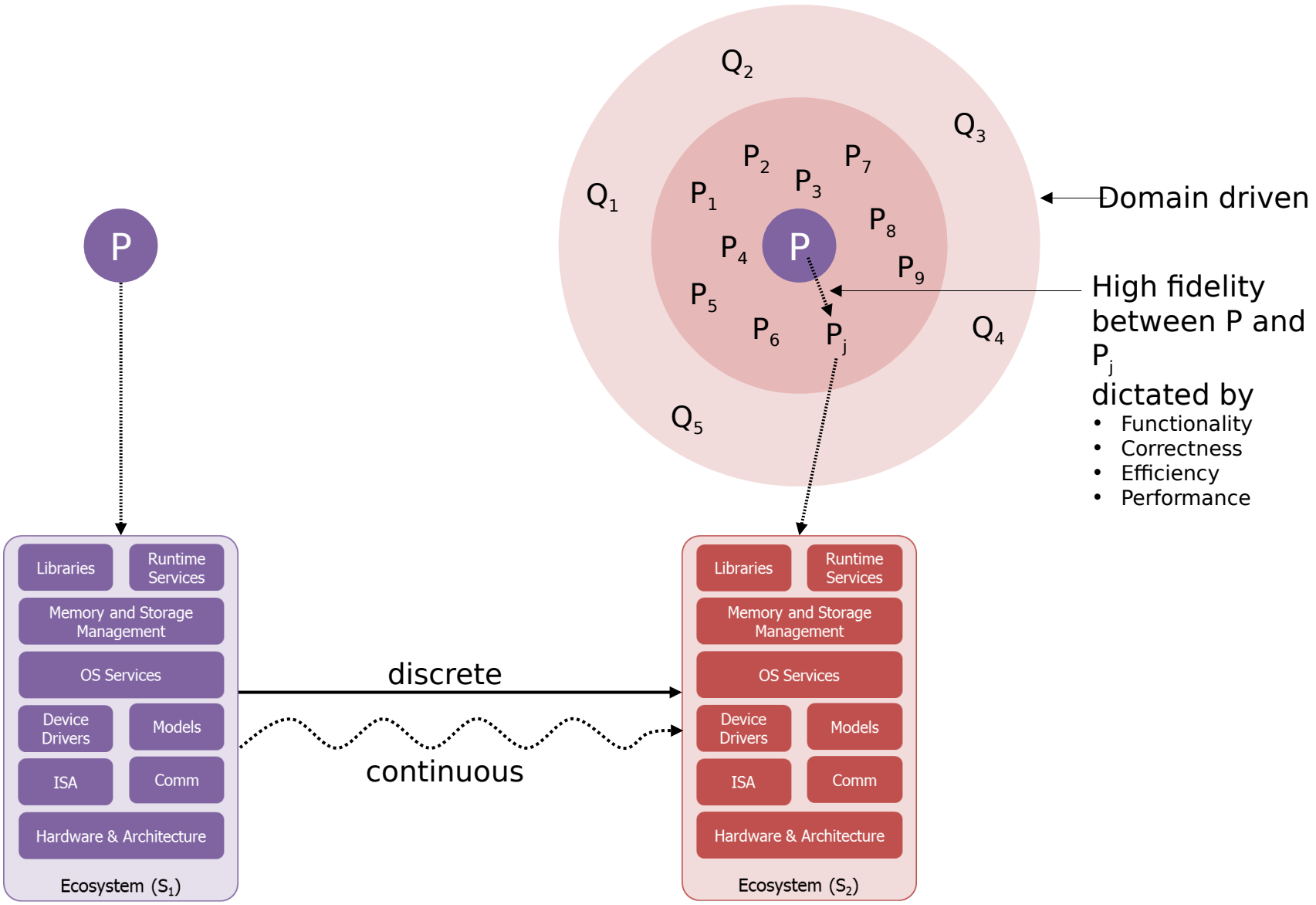
Altered Functionality

- Computation and memory models
- Data formats

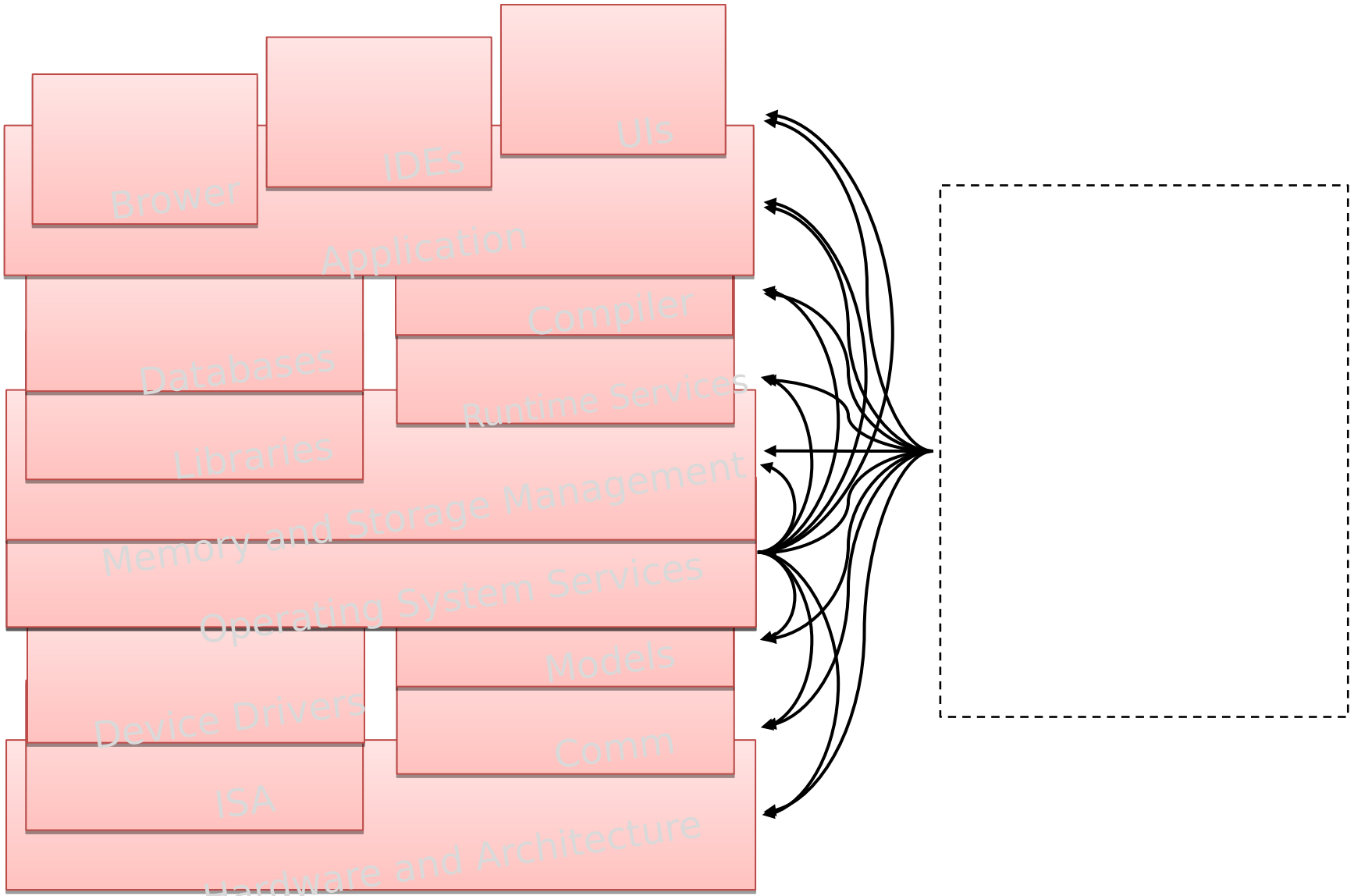
Goal



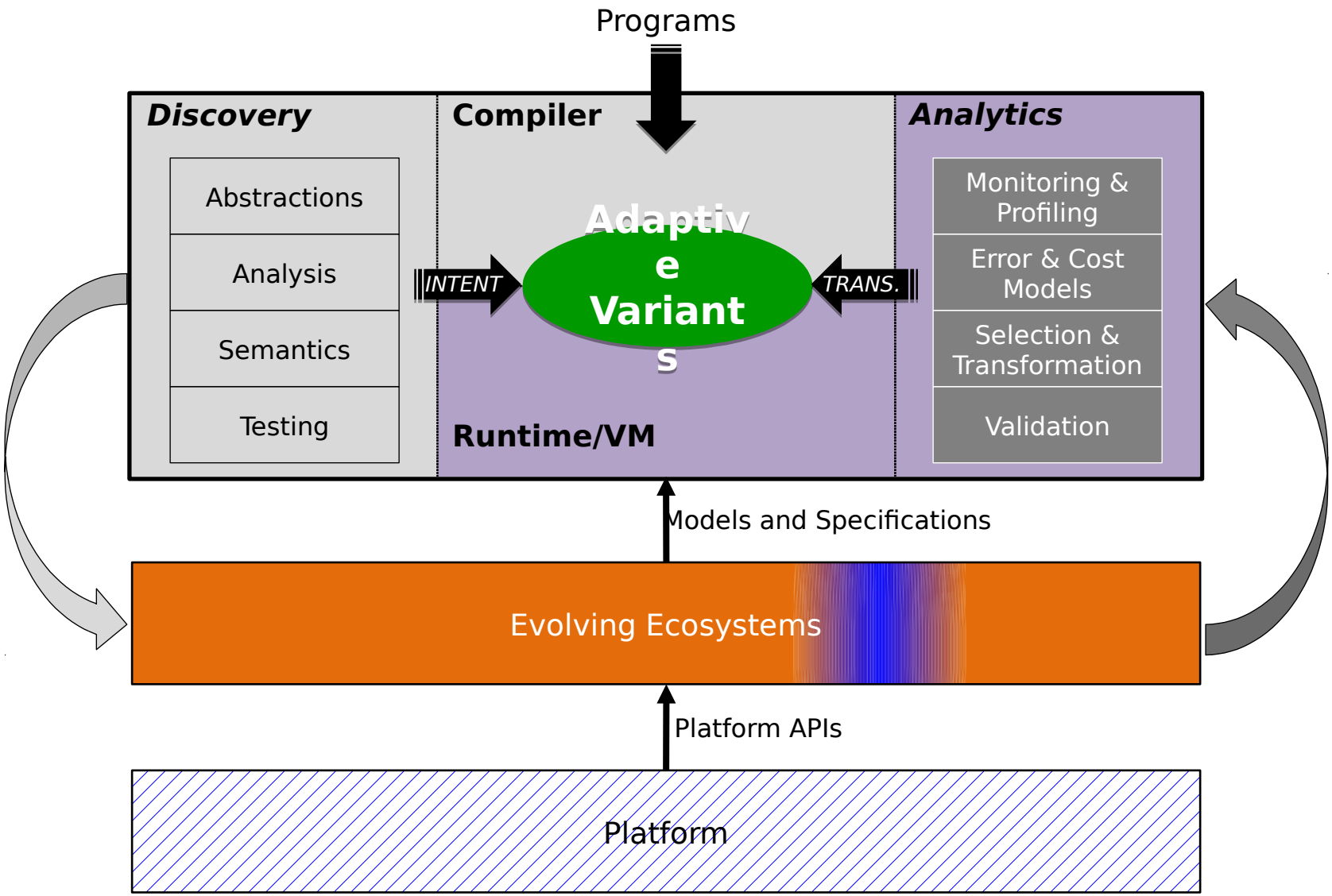
Idea



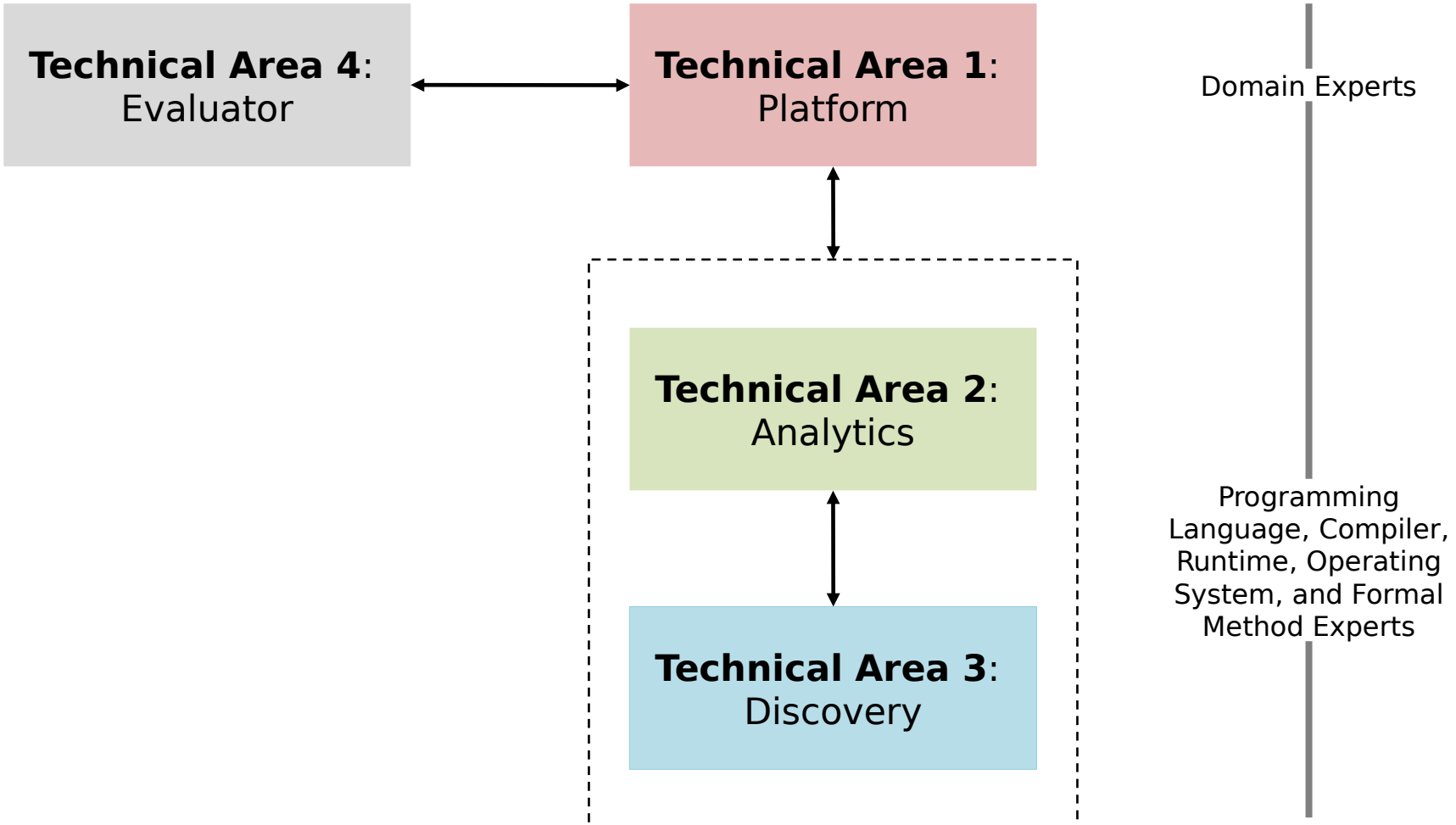
Idea


UIs
IDEs
Brower
Application
Compiler
Databases
Runtime Services
Libraries
Memory and Storage Management
Operating System Services
Models
Device Drivers
Comm
ISA
Hardware and Architecture

Architecture



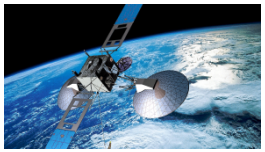
Program Structure



TA1: Platform

Exemplar platforms include, but are not limited to

- Autonomous and robotic systems
- Embedded systems (e.g., Internet of Things)
- Geo-distributed systems
- Heterogeneous scalable multiprocessors
- Cloud infrastructure
- Mobile computing
- High-assurance systems
- Coordinated platform (i.e., Swarm)
- Storage and file systems
- Diverse hardware infrastructures (e.g., FPGA)


Source: darivoa.com


Source: www.forum.eideha.com

Source: adrotelecom.nl


Source: cs.cmu.edu


Source: ibm.com


Source: flagvruki.com


Source: HACMS

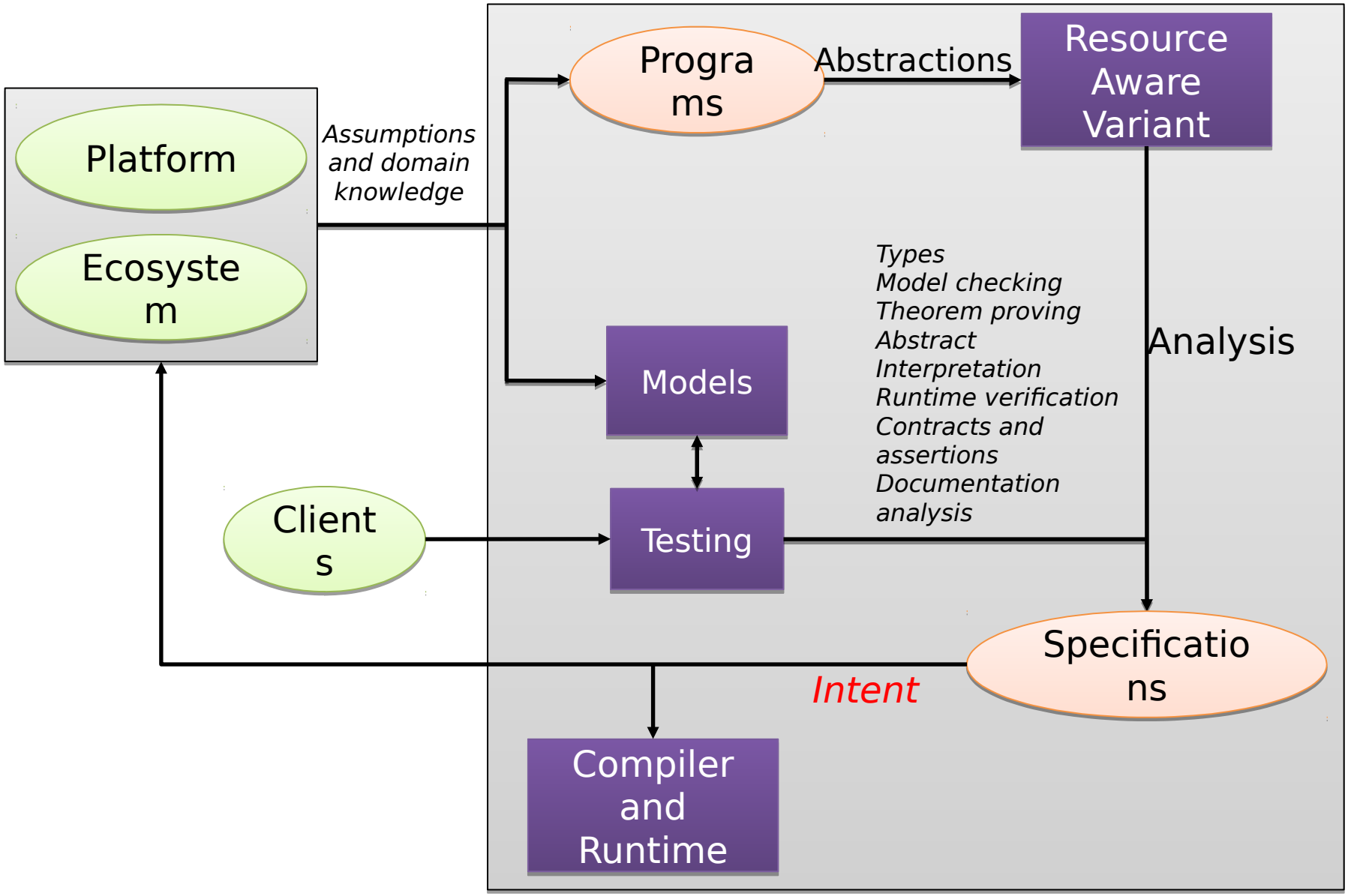

Source: cisco.com


Source: eurotech.com

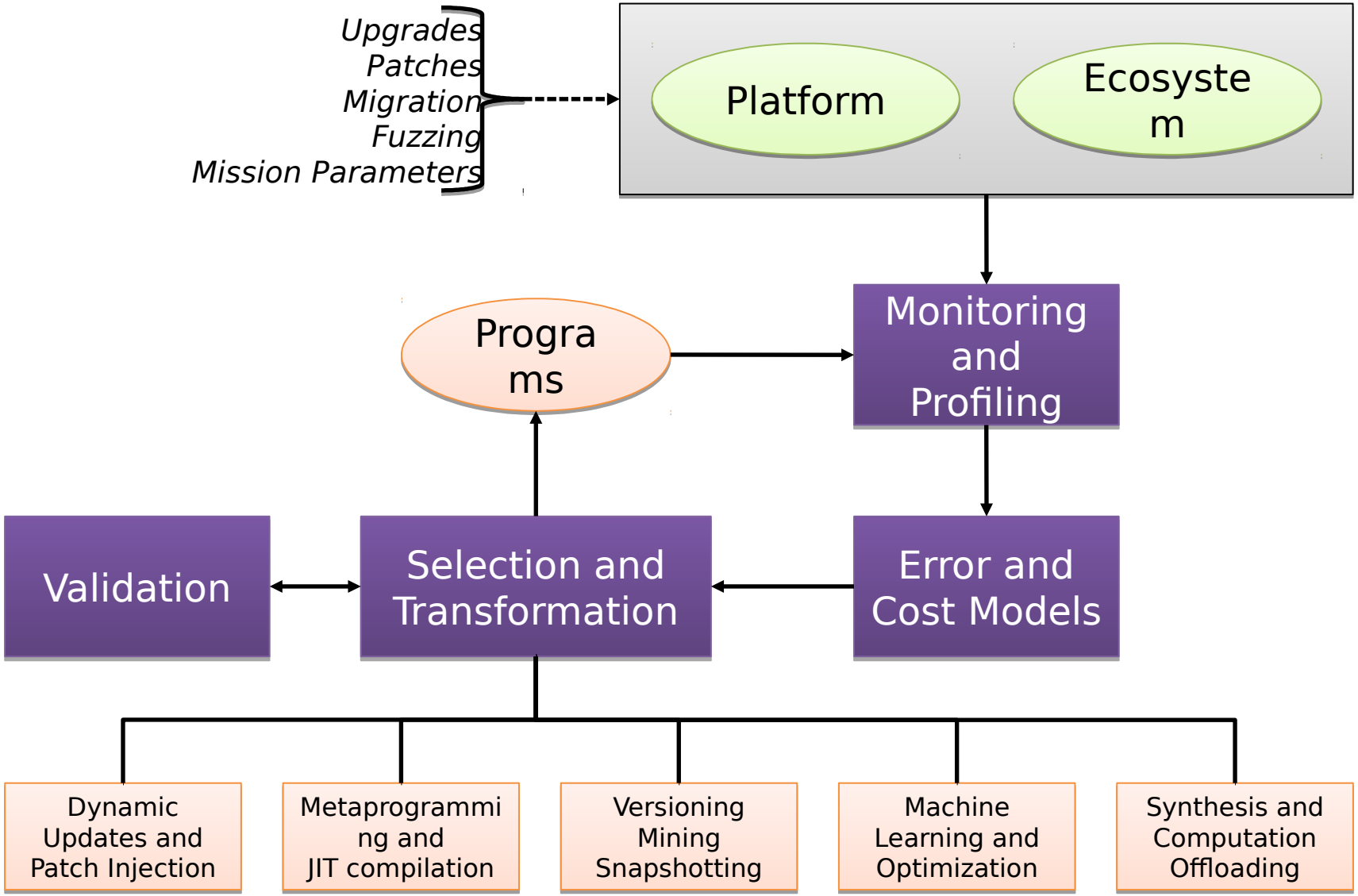
Source: blog.randahl.dk

Source: reusetek.com

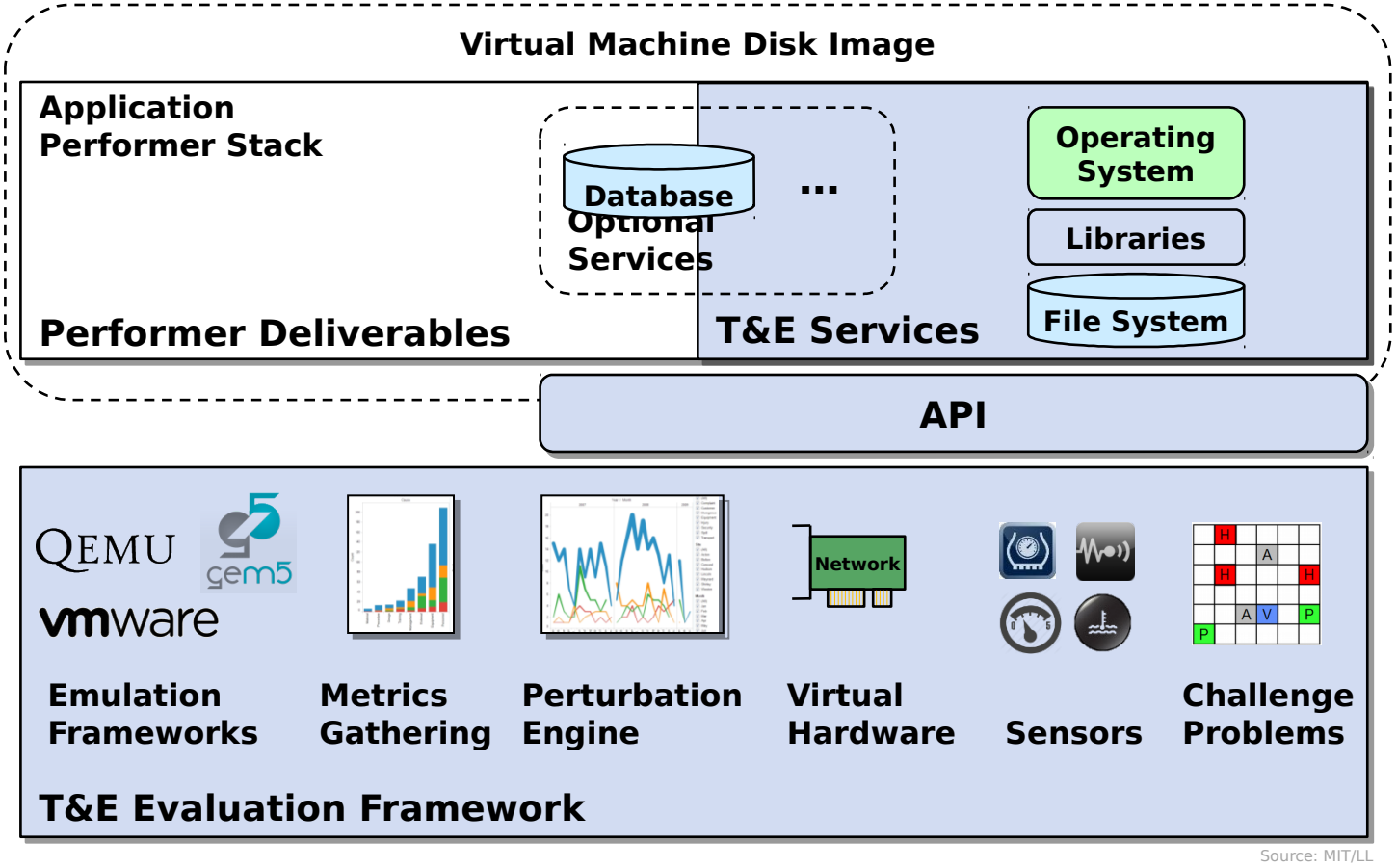
TA3: Adaptive Front-End *Discovery*



TA2: Adaptive Back-End *Analytics*

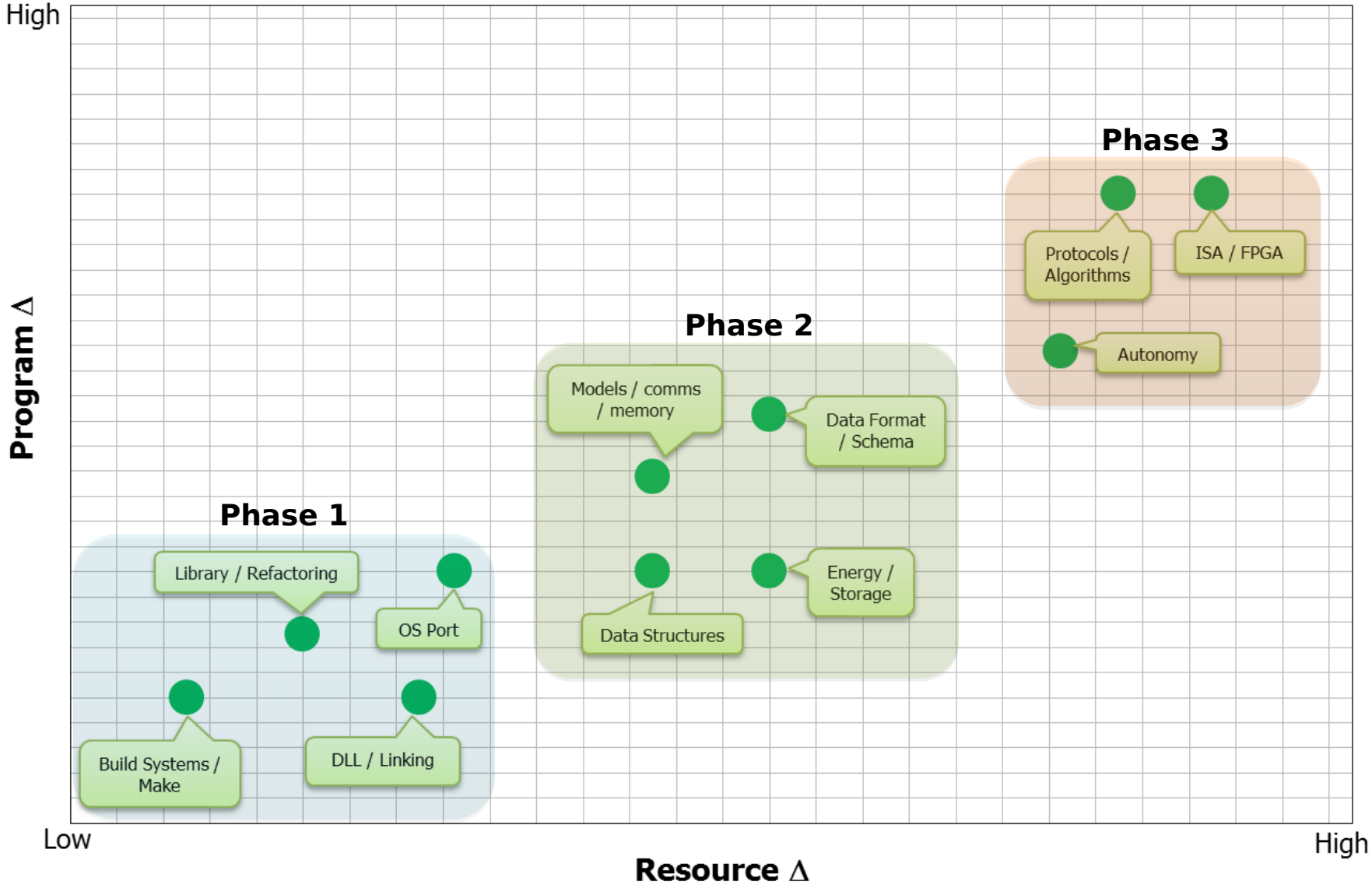


TA4: Evaluator (*not solicited*)

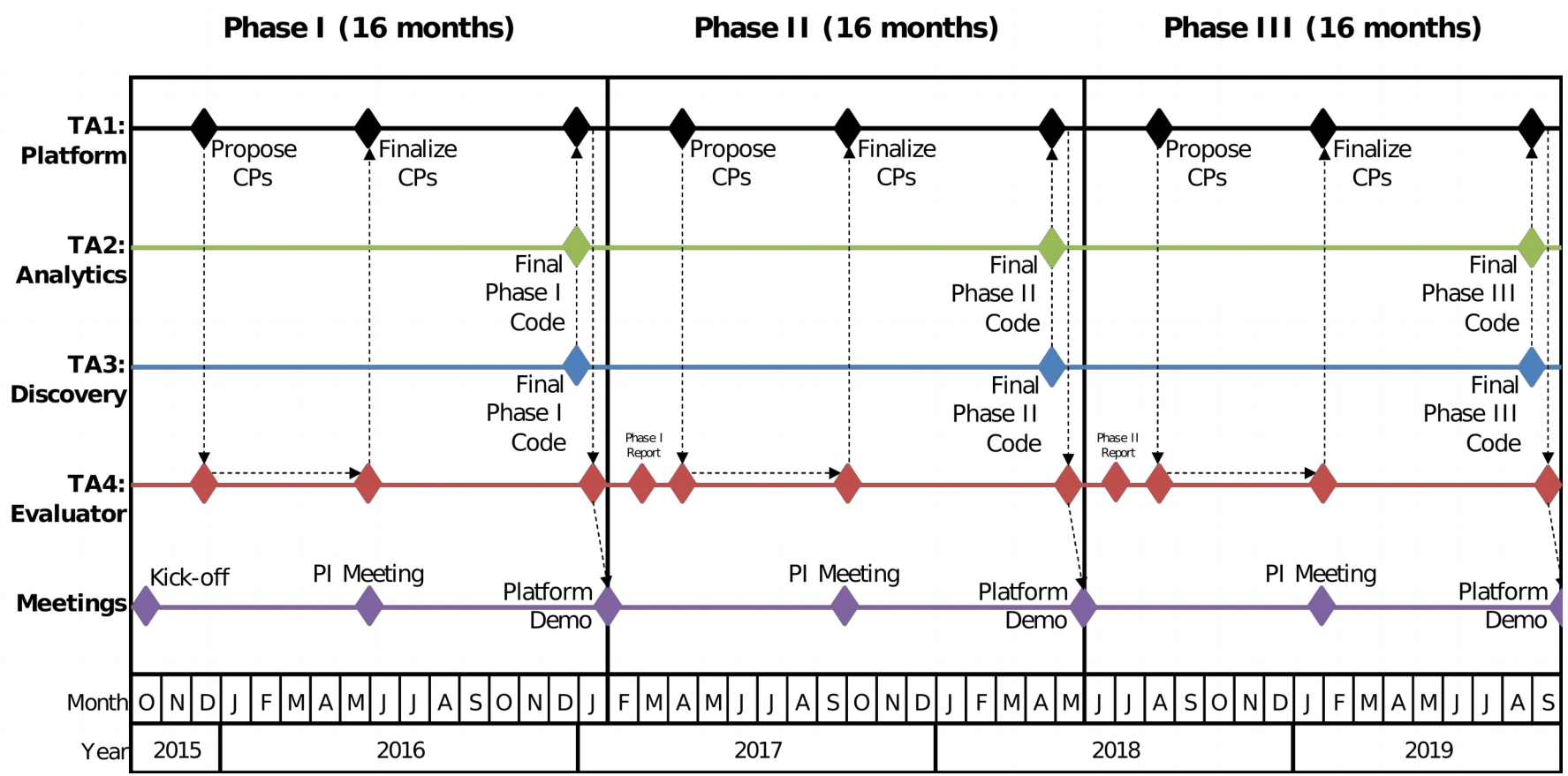


*Notional

If this technology is wildly successful…



Schedule, Products, & Transition Plan



Products

- Fully adaptive platforms
- Adaptive monitoring and transformation tools
- Continuous resource adaptive analysis suite

Transition Plan

- Include Government customers involved in related domains on challenge program selection and encourage participation in “Platform Demos”.

Overall Programmatics

- Three 16 month Phases
- Four Technical Areas (TAs)
  - TA1 - Platform
  - TA2 - Analytics
  - TA3 - Discovery
  - TA4 - Evaluator (not solicited)
- Anticipate multiple awards in TAs 1, 2, and 3
  - The TA4 (Evaluator) will be provided by the Government and proposals for this area will not be solicited under this announcement
- Anticipated award amount for an integrated TA1-3 effort will range between $8-$10M
- Performers in TA1-3 will be grouped into one design team
  - Each team led by a TA1 performer
  - Each team will have one TA2 performer and one or more TA3 performers
  - Each team will produce a working end-to-end Discovery and Analytics System (DAS)
  - Teams will not be competitively evaluated; no anticipated down-selection

TA1 Programmatics

- Lead a DAS design team
  - One TA2 performer and one or more TA3 performers
  - Produce a working end-to-end DAS
  - Provide Platform/Domain knowledge to DAS design team

- Interaction with TA2 and TA3 performers
  - Educate them about the technical challenges in producing resource adaptive versions of the platform
  - Develop unrestricted and unclassified Platform-Specific Challenge Problems
  - Apply their team's research results to the development of a working end-to-end DAS
  - Coordinate and finalize definitions of APIs, interfaces, representations, internal DSLs, and other technologies necessary to enable interaction among their team

- Interaction with the Evaluator
  - Propose two Platform-Specific Challenge Problems two months after the program kick-off and three Platform-Specific Challenge Problems two months after the start of Phase 2 and Phase 3
  - Finalize the Platform-Specific Challenge Problems by the PI meeting in each Phase
  - Deliver the DAS for evaluation two weeks prior to each Platform Demonstration Workshop
  - Serve as the primary POC for technical support during the evaluation of the DAS
  - Demonstrate the DAS on the Platform-Specific Challenge Problems at each Platform Demonstration Workshop
  - Initiate discussions on proposed Platform-Specific Challenge Problems for the following phase
  - Work with the Evaluator to define an API between the DAS and the Evaluator's evaluation framework

- It is strongly preferred that the identified platforms, including sensors, computing hardware, and software, are unclassified

TA2/TA3 Programmatics

- Participate in one DAS design team, led by a TA1 performer
- Use the Platform-Specific Challenge Problems developed by the partnered TA1 performer to evaluate and drive research
- Deliver appropriate code and documentation to their partnered TA1 performer at regular intervals, and on a schedule consistent with the delivery of a working DAS three weeks before each Platform Demonstration Workshop
- Work with performers from all TAs to identify critical research challenges and issues
- *TA2 and TA3 proposals that are not part of a collaborative TA1 effort must nonetheless clearly justify the utility of their approach in the framework of a potential platform.*